CompSci 162
Spring 2023 Lecture 18:
Reducibility

Review

- Turing Recognizable and co-Turing Recognizable

both: Turing decidable

- Barber Paradox:
  - A town has exactly one barber
  - The barber cuts the hair of exactly whoever does not cut their own hair.
  - Who cuts the barber's hair?

Does this Turing Machine Accept?

$A_{TM} = \{\langle M, w\rangle : M \text{ is a Turing Machine and } M \text{ accepts } w\}$

- This language is a set of strings
  - Every string in the language is a TM and a string, such that if you run that TM on that string, the result is the TM accepts.
- This is **undecidable**
- Suppose it were decidable. Then H decides it
- But then I could build D
  - Input: TM M
  - Behavior: If $H(M, M)$ accepts, reject
    If $H(M, M)$ rejects, [b/c reject or b/c or loops forever], accept
- What happens if I call $D(D)$?

The Halting Problem

$HALT_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on input } w\}$

- Suppose FSOC $HALT_{TM}$ is decidable.
- Then $\exists$ TM R that decides ~~it~~. $HALT_{TM}$
- // Use R to create S, which decides A_{TM} (which we know is undecidable)

Run R on $\langle M, w\rangle$

if R rejects $\langle M, w\rangle$, S rejects $\langle M, w\rangle$

else run M on input w

If M accepts w, S accepts $\langle M, w\rangle$

else S rejects

E_{TM} is undecidable

$E_{TM} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \emptyset\}$
Suppose FSOC that E_{TM} is decidable and I want to decide A_{TM} given input $\langle M, w \rangle$

- Let R be the TM that decides E_{TM}
- // Use R to create S to decide A_{TM} (which we know is undecidable)

- We could run R on M
- If accept, $L(M)$ is empty.
- If reject, we don't know if M accepts w

E_{TM} is undecidable

$E_{TM} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \emptyset\}$
Suppose FSOC that E_{TM} is decidable and I want to decide A_{TM} given input $\langle M, w \rangle$

M' accepts either nothing OR just w

- Let R be the TM that decides E_{TM}
- // Use R to create S to decide A_{TM} (which we know is undecidable)
- Create M' which has input x
  - If $x \neq w$, reject // hard coded
  - Else run M on w

Caution: do not run M'

- Run R on M'
  If R accepts: A_{TM} reject $\langle M, w \rangle$
  else A_{TM} accepts $\langle M, w \rangle$

EQ_{TM} is undecidable

$EQ_{TM} = \{\langle M_1, M_2\rangle :$
M_1 and M_2 are TMs and $L(M_1) = L(M_2)\}$

General strategy: Undecidable proof

Problem: prove X is undecidable

- Suppose I had a TM that decides X
- Pick an undecidable problem Y
- Write a TM to decide Y
  - Must be a valid TM **EXCEPT**
    it assumes existence of TM to decide X
- But that would decide Y
- By contradiction, no TM for X

I consider "wrong direction" to be major error